miniSlicer UI 翻译流水线技术报告

自动生成

May 11, 2025

1 场景与目标

- **场景**：miniSlicer 项目包含数千个 Qt Designer 生成的 `.ui` 文件，需要将界面文字由英文批量翻译为中文。
- **目标**：
  1. 在**全离线 / 本地仓库**条件下完成文本抽取、调用 LLM 翻译、写回文件，并保证 XML 格式合法。
  2. **最小侵入**：原始文件仅在最后一步被覆盖；中间过程使用临时目录，方便回滚。
  3. **可持续迭代**：脚本可重复执行，支持后续 UI 变动的增量翻译；流水线易复用到其他格式或项目。

2 总体流程

流水线由五个核心脚本组成，见表 1，执行顺序如同一条数据流：

Table 1: UI 翻译流水线各步骤

序	入口脚本	核心职责
1	`generate_ui_placeholders.py`	递归扫描 `.ui` 文件，对需要翻译的 `<string>` 节点加上占位符 `¥0000000123¥` 并复制到 `tmp_ui/`；同时拼接大文件 `all_strings_tagged.txt` 供 LLM 输入。
2	`translate_ui_with_openai.py`	以 System Prompt + 大文件分块调用 OpenAI（可指定 `--base_url` 代理），生成 sed 风格的替换脚本 `replace.sh`。
3	`apply_xml_translations.py`	解析 `replace.sh`，将中文文本填充回 `tmp_ui/` 下对应节点，并按原相对路径写到源码目录。
4	`remove_ui_placeholders.py`	在 `tmp_ui/` 阶段去掉占位符后缀，保证写回的 XML 已无标识符。

5	fix_ui_placeholders.py	兜底：在真正源码树再扫一次，防止极端情况残留占位符。

Makefile 目标

```
$ make -C script all      # = translate
$ make -C script clean    # 删除 tmp_ui 等临时产物
$ make -C script monitor  # 查看 OpenAI token 等
$ make -C script fix      # 单独运行最后修复步骤
$ make -C script code_translate  # 源码 title/tr 等翻译
$ make -C script code_fix        # 清理 UI+源码占位符
```

3 关键技术拆解

3.1 占位符标记策略

- 使用稀有字符"¥" 包围，固定宽度 ID_WIDTH=10，形如 ¥0000000123¥ 。
- 正则匹配简单：^¥\d{10}¥\s? 能快速定位。
- 与 UTF-8 中文不冲突；id_map.jsonl 记录 *ID*–路径映射，方便调试。

3.2 LLM 翻译交互

- **Prompt Engineering**：系统提示固定，强调医学影像软件本地化风格。
- **块切分**：环境变量 CHUNK_SIZE 控制单次输入长度，脚本自动拆分与合并。
- **安全写回**：不直接 shell sed，而是解析为字典后用 XML DOM 写入，避免格式损坏。

3.3 结构化写回

- 统一使用 xml.etree.ElementTree 读写，保持属性顺序。
- 仅对变更节点写文件，减少无意义 diff。

3.4 校验与回滚

- 若本地安装 xmllint，自动校验所有生成的 .ui 文件。
- tmp_ui/ 保留带占位符的中间文件，可回滚。
- fix_ui_placeholders.py 作为最后保险，确保仓库无遗留占位符。

4 方法论抽象

以下模式可迁移到任何结构化批量翻译或替换任务：

1. **可逆标记（Tagging）**：为文本加唯一占位符，建立映射。
2. **大文件拼接（Concatenate）**：将所有带标记文件拼成 LLM 输入，解决上下文与截断问题。
3. **LLM 翻译（Translate）**：分块调用模型，统一输出格式（如 sed/JSON）。
4. **结构化应用（Apply）**：解析替换结果，遍历原文件结构精准写回。
5. **标记清理（Cleanup）**：移除占位符，必要时全局扫描兜底。
6. **校验与监控（Verify & Monitor）**：格式校验器、token 监控，保障质量与成本可见性。

5 可迁移示例

场景	文件格式	标记节点示例
Vue i18n	`.vue` / `.ts`	`t('¥0000001¥ Hello')`
JSON 配置	`.json`	`"title": "¥0000023¥ My Title"`
Markdown 文档	`.md`	`> ¥0000042¥ Some sentence`
Slicer XML 描述	`.xml`	同本案例

6 后续改进方向

1. **增量翻译**：借助 `git diff` 仅重标记新增英文文本。
2. **Prompt 动态优化**：根据历史翻译质量自动调整 style guide。
3. **多语言支持**：占位符阶段记录目标语言代码，支持一次生成多语种。
4. **CI 集成**：在 PR workflow 中自动执行 `make translate` 并回传结果。

7 更多应用设想

除本报告所述 UI 翻译外，该批量标记–清洗–回写的方法论可延伸至更广阔的场景，列举如下，供脑洞参考：

1. **自动脱敏 / 隐私保护**[1]：对企业文档批量插入占位符以标记姓名、地址、身份证号等敏感实体，随后调用 LLM 进行假名化或消除，再写回原格式（Word/Markdown/PDF 元数据）。

[1]如 GDPR 合规

2. **医疗影像 DICOM 头一致化**：批量扫描 DICOM 文件，使用占位符标记医院、日期等字段，经脚本统一重写以满足多中心数据集匿名化与字段映射。

3. **金融报表数字校验**：对 Excel/CSV 批量添加占位符，LLM 校对币种、千分位与四舍五入规则，再写回清洗数据。

4. **软件遗留代码现代化**：在上百万行 C++/Python 代码中占位符标记过时 API，借助 LLM 给出新 API 替换语句，自动补丁回写并生成迁移报告。

5. **合规术语统一**：大型企业内部多语种政策文件，通过占位符标注术语，利用 LLM 统一术语翻译并维护术语库。

6. **游戏文本本地化 + 文化化**[2]：RPG 游戏剧情脚本加占位符，LLM 根据地区文化自动改写梗、换装置之后写回脚本文件。

7. **多模态字幕同步**：电影字幕 `.srt` 文件中先插标记，再用 LLM 将翻译与时轴长度匹配（字符数压缩），回写保证口型同步。

8. **日志级别重构**：在微服务日志中占位符标记 INFO/DEBUG 字符串，LLM 根据运行频率与运维策略调整为 WARN/ERROR 等级并回写配置。

9. **知识库 SEO 优化**：批量占位符插入网页标题与 meta 描述，LLM 生成关键词优化后的文本并回写 HTML/Markdown。

10. **VR/AR 场景提示词转换**：对互动脚本批量标记交互提示，LLM 根据设备（VR/AR/移动）自动改写提示词，保持沉浸感。

11. **跨平台快捷键映射**：批量占位符标记 Windows/Mac 键位提示，LLM 根据目标平台自动转换为 Cmd/Control 等符号并回写帮助文档。

12. **论文参考文献格式统一**：在 BibTeX/EndNote 导出中插入占位符，LLM 统一期刊缩写与 DOI 链接，回写生成期刊规范格式。

13. **API 文档示例代码多语言生成**：占位符标记 Python 示例，LLM 生成 JavaScript/Go 等语言对等示例并插入到 Markdown 文档中。

14. **智能重写 commit message**：批量标记简短 commit，LLM 依据 Conventional Commits 规范生成更具信息量的描述并回写 git 历史（rebase）。

15. **IoT 设备配置国际化**：在大规模设备固件 JSON 中插入占位符，LLM 将用户提示翻译为当地语言并考虑单位制转换。

16. **合同条款风险扫描与替换**：标记潜在高风险条款，LLM 给出更友好措辞或提示并将修订版插回 Word/Markdown 合同。

17. **曲谱移调与和弦替换**：在 MusicXML 文件标记和弦序列，LLM 根据歌手音域自动移调与简化和弦后写回曲谱。

[2]Culturalization

18. **化学分子命名标准化**：在科研数据 CSV 中占位符标记分子俗名，LLM 替换为 IUPAC 系统命名。

19. **OpenAPI 版本升级**：批量标记旧版 Swagger YAML 中弃用字段，LLM 生成 v3 等效字段并迁移示例。

20. **教育测验题干难度调节**：在题库 JSON 标记题干，LLM 根据 Bloom 认知层级重写题目以生成多难度版本。

21. **社交媒体敏感词过滤**：占位符标记潜在敏感词汇，LLM 根据地区法律自动替换或打码。

22. **古籍 OCR 纠错**：在批量 OCR 结果中占位符标记低置信度字符，LLM 参考上下文进行智能纠错。

23. **电商商品标题优化**：在商品 CSV 标题列插入占位符，LLM 生成长尾关键词并保持字符限制。

24. **机器学习特征命名规范化**：在代码与配置中占位符标记特征名，LLM 统一蛇形/驼峰命名并同步到文档。

25. **变更日志自动分类**：批量标记 release note 行，LLM 根据语义分类为 Feature/Bugfix/Perf 等，并输出结构化 CHANGELOG。

26. **多语种字幕情绪标注**：在剧集字幕标记句子，LLM 推断情绪标签（喜/怒/哀）并作为嵌入写回。

27. **3D 打印 G-code 参数优化**：占位符标记关键速度/温度指令，LLM 根据材料建议参数，批量替换生成新 G-code。

28. **CLI 帮助信息一致化**：在各微服务 CLI 输出占位符标记帮助文本，LLM 统一格式与示例。

29. **多地区法律条款对照**：在法规数据库占位符标记条款编号，LLM 生成跨国对照摘要并回写 JSON。

30. **网站可达性增强**：批量标记 HTML img 缺失 alt 属性，LLM 根据上下文生成描述并自动补全。

8 源码字符串翻译流水线

本节说明如何借助前一章的“占位符 →LLM → 回写”范式，把 **Python / C++ 源码中的可翻译字符串**（模块 title、Qt `tr()` 等）批量翻译成中文。

8.1 新增脚本说明

脚本	作用
generate_code_placeholders.py	递归扫描指定后缀（.py .c .cpp .h ...），匹配三类正则：_('text')、self.parent.title = 'text'、::tr("text")，插入占位符并生成 tmp_code/ 与 all_strings_code.txt。
apply_code_translations.py	解析 LLM 输出的 replace.sh，将中文写回 tmp_code/，随后同步到源码目录。
fix_placeholders.py	通用占位符清除器；支持 --ext，自适应 XML 或纯文本。

8.2 Makefile 一键命令

```
$ make -C script code_translate    # 插入占位符 -> LLM 翻译 -> 写回源码
$ make -C script code_fix          # 清理 UI 与源码中残留的 ¥XXXXXXXXXX¥ 占位符
```

脚本接受与 UI 流水线相同的环境变量 MODEL/CHUNK_SIZE/BASE_URL，可直接在 CI/CD 环境注入。

8.3 参数化与可扩展性

- **可扩展正则**：通过 --pattern "(?P<prefix>....)(?P<text>...)(?P<suffix>...)" 追加匹配模式；命名捕获组 text 为必需。
- **多后缀支持**：--ext py,rs,go 可轻松覆盖 Rust、Go 等新语言。
- **代理与分块**：沿用 UI 流水线的 --base_url 与 --chunk_size，大文件可按需拆分。

此后若 UI 与源码均需翻译，可先执行 make translate 完成 UI 部分，再运行 make code_translate 同步源码文本；最终用 make code_fix 统一清除占位符。